Bibliografie

1. Andrew S. Tanenbaum – „Organizarea structurata a calculatoarelor “ – Editia a IVa, Ed. Agora, Tg. Mures, 1999.
2. Miles J. Murdocca, Vincent P. Heuring – „Principles of Computer Arhitecture”, Prentice – Hall, 1999.
3. Willian Stallings – „Computer Organization and Architecture”, Prebtice-Hall, Inc., 2000.
4. Peter Norton – “Sectrete PC”, Ed. Teora, Bucuresti 1998(traducere SAMS Publishing, 1995).
5. Radu Mârsanu –“Calculatoare persoanale-elemente arhitecturale”, Ed. BIC ALL, Bucuresti, 2001.
6. Barry B. Brey– „The Intel Microprocessors”, Fifth Edition, Prentice-Hall, Inc., 2000.
7. Peter Abel – „IBM PC Assambly Language and Programming”, Fifth Edition, Prentice-Hall, Inc., 2001.
8. Baruch Zoltan Francisc - "Sisteme de intrare–iesire ale calculatoarelor” Ed. Albatros, Cluj – Napoca, 2000.
9. Vijay K. Madisetti – “Digital Signal Processors”, Butterworth-Heinemann, 1995.

I Introducere

Arhitectura calculatoarelor trateaza comportarea functionala a unui calculator asa cum este vazut acesta de catre programator. Acest punct de vedere include aspecte ca marimea si tipul datelor (deexemplu folosirea a 16 biti pentru a reprezenta un întreg) si tipul operatiilor pe care le suporta sistemul (ca aditie, substractie, call subrutina).
Organizarea calculatorului trateaza relatia structurala, care nu este vizibila pentru programator, între elementele sistemului (de ex. interfetele cu dispozitivele periferice, tipul si tehnologia folosita pentru memorii etc.)
Cursul va trata atât arhitectura cât si organizarea calculatoarelor prin termenul de “arhitectura’ facându-se referire la ambele, arhitectura si organizare.
Exista conceptul de nivele în arhitectura calculatorului. Ideea de baza este aceea ca exista mai multe nivele de pe care un calculator poate fi privit, de la nivelul cel mai înalt, de la care utilizatorul ruleaza programe, pâna la nivelul cel mai de jos, constând din tranzistoare, rezistente si fire.
Înainte de a discuta aceste nivele, vom trece în revista un scurt istoric al sistemelor de calcul pentru a avea o perspectiva a evolutiei în timp a acestora.

1.1 Scurt istoric

Au existat unele dispozitive mecanice pentru controlul operatiilor complexe înca din anii 1500, când cilindrii rotitori cu cama erau folositi pentru cutii muzicale, asa cum le cunoastem astazi.

Blaise Pascal (1923–1662) a construit o masina de calcul functionala pe când avea doar 19 ani (1642) pentru a-l ajuta pe tatal sau, colector de taxe al guvernului francez. Masina era în întregime mecanica si putea sa faca numai adunari si scaderi si exista si în ziua de astazi.
Treizeci de ani dupa aparitia calculatorului "Pascaline", Gottfried Willelm von Leibnitz (1646 – 1716) a construit o alta masina masina mecanica compatibila sa faca înmultiri si împartiri. De fapt, Leibnitz a reusit sa construiasca echivalentul unui calculator de buzunar cu patru operatii, cu trei secole în urma.
Charles Babbage (1792–1871), prof de matematica la Universitatea din Cambridge, inventatorul vitezometrului, a proiectat si a construit masina de calcul a diferentelor (the difference engine). De fapt, el nu a construit niciodata o varianta practica a masinii pe care o proiectase. Babbage a trait în Anglia în perioada în care tabelele matematice erau utilizate în navigatie si activitati stiintifice. Masina a fost proiectata sa faca doar adunari si scaderi. Tabelele erau calculate manual si contineau erori. Scopul a fost de a crea o masina care sa poata calcula tabele folosind un singur algoritm, si anume, metoda diferentelor finite folosind polinoame. Aceasta masina avea si un dispozitiv de iesire-stanta rezultatul-fiind astfel un procursor al mediilor periferice unic inscriptionabile de mai târziu (cartele perforate sau CD-ROM-uri). Succesul acestei masini l-a determinat sa se dedice unei alte masini-analytical engine-care a fost construita si cu sprijinul guvernului. Aceasta poseda un dispozitiv pentru "branching" (luarea deciziilor) si mijloace pentru programare. Masina analitica avea 4 componente: magazia (memoria), moara (unitatea de calcul), sectiunea de intrare (cititorul de cartele perforate) si sectiunea de iesire (iesirea perforata si imprimata). Masina citea instructiunile de pe banda perforata si le executa. De exemplu puteau fi extrase doua numere din magazie, sa fie aduse în moara si prelucrate (adunate de exemplu) si sa trimita rezultatul din nou în magazie. Putea efectua salt conditionat în functie de felul numarului-pozitiv sau negativ. Prin perforarea a diferite programe, masina putea efectua diverse calcule (lim, $\sqrt{}$, etc). Deoarece masina era programabila (magazia avea 1000 cuvinte a câte 50 de cifre zecimale) avea nevoie de software. Pentru a produce acest software Babbbage a angajat o tânara, Ada Augusta Lovelace, fica poetului Lord Byrom. Astfel Ada Lovelace a devenit primul programator de calculatoare din lume. Limbajul Ada a fost numit astfel în onoarea ei.
Marele dezavantaj al acestei masini era acela ca nu a putut fi pusa la punct în întregime datorita tolerantelor mecanice imposibil de realizat cu tehnologia secolului XIX.

La sfârsitul anilor '30, un student german, Konrad Zusse, a construit o serie de masini de calcul automat folosind relee electromagnetice. Zusse nu a cunoscut masinile lui Babbage si realizarile sale au fost distruse în timpul bombardamentelor aliatilor asupra Berlinului în 1944, deci munca lui nu a influentat masinile ce au urmat.

În aceasi perioada au mai existat si alte proiecte ale unor masini de calcul cu relee.

- John Atanasoff – de la State College of Iowa,
- George Stiblitz – de la Bell Laboratories,
- Howard Aiken – de la Harvard – Mark I – 1944. Când a finalizat Mark II calculatoarele cu relee erau deja depasite.

1.1.1 Prima generatie – tuburile electronice

În timpul celui de al doilea razboi mondial, submarinele germane (U – boot-urile) dispuneau de un sistem de comunicatie radio cu un cod criptat numit ENIGMA. Serviciile secrete britanice au obtinut o masina ENIGMA de la polonezi la începutul razboilui. Procesul de codare a mesajelor

criptate (codificate) era anevoios deoarece necesita un volum enorm de calcule si, pentru a putea fi utilizate, acestea trebuiau efectuate la scurt timp de la interceptarea mesajului. Problema a fost în final rezolvata cu ajutorul matematicianului Alan Turing (1912–1954). Pentru a solutiona problema decodificarii, guvernul britanic a construit la Bletchway Park primul calculator numeric electronic din lume. Alan Turing a contribuit la proiectarea acestei masini, care a devenit operationala în 1943 si a fost numita COLOSSEUM.
Între timp, în Statele Unite, John Mauchley si-a dat seama ca armata este interesata de sisteme de calcul. El a facut o propunere de grant prin care cerea fonduri pentru construirea unui calculator capabil sa calculeze tabele pentru traiectoriile balistice pentru armata SUA. În 1943 John Mauchley, împreuna cu asistentul sau J. Presper Eckert au început sa construiasca un calculator pe care l-au numit ENIAC (Electronic Numerical Integrator and Computer). Acesta cântarea 30t si consuma 140Kw. ENIAC era programat prin 6000 de comutatoare multi–pozitionale. Masina a fost terminata în 1946.

Eckert si Mauchley, care erau la Moore School at the University of Pennsylvamia, au început sa lucreze la un alt calculator pe care l-au numit EDVAC (Electronic Discrete Variable Automatic Computer). Proiectul a fost compromis datorita faptului ca si-au creat propria companie in Philadelphia,
Între timp, unul din cei ce lucrase împreuna cu la ENIAC, John von Neumann (1903-1957), la Institute for Advanced Studies at Princeton, a trecut la construirea versiunii propii a EDVAC – ului, IAS. EDVAC si IAS au fost pimele computere cu program stocat. Primul model functional de calculator cu program stocat a fost EDSAC, construit de Maurice Walkes la Cambridge University 1947 si operational în 1949.

Modelul vom Neumann

Calculatoarele conventionale digitale au o forma comuna care îi este atribuita lui ***vom Neumann,*** dar istoria reala ca o întreaga echipa a fost responsabila pentru elaborarea acestui model.
Modelul ***vom Neumann*** consta din 5 componente ca în figura 1.

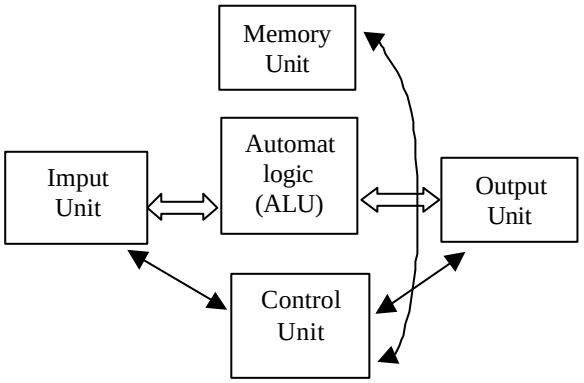

Fig. 1 Modelul vom Neumann al calculatorului numeric

Input Unit si Output Unit formeaza grupul dispozitivelor de intrare iesire ce asigura instructiuni si date sistemului, care anterior au fost stocate în Memory Unit.

Instructiunile si datele sunt procesate de ALU, sub controlul unitatii Control Unit. Aceasta determina executia instructiunilor una dupa alta, asa cum au fost ele memorate, atât timp cât aceasta secventa nu este întrerupta de o instructiune de salt.
ALU + CU formeaza unitatea centrala (CPU).
Programul stocat sau ***memorat*** este cel mai important aspect al masinii ***vom Neumann.*** Un program este stocat în memoria calculatorului împreuna cu datele. Desi în prezent cunoastem acest fapt ca un dat, anterior dezvoltarii calculatoarelor cu program stocat, programele erau memorate pe cartele, benzi perforate sau benzi magnetice. În calculatoarele cu program memorat, programul poate fi manipulat ca si datele. Acest fapt a dat nastere compilatoarelor si sistemelor de operare si a facut posibila marea versabilitate pe care o au calculatoarele actuale.

1.1.2. A doua generatie – tranzistoarele (1955 – 1965).

Tranzistorul a fost inventat la Bel Laboratories în 1948 de John Bardeen, Walter Brattain si William Shockley.
Primul calculator cu tranzistoare a fost construit în laboratorul Lincoln al M.I.T. si a fost numit TX–0 (Transistored experimental computer 0). Urmatorul model a fost TX-2. Unul din inginerii care a lucrat la TX-2, Keneth Olsen, a format în 1957 o companie numita Digital Equipment Comparation (DEC). Primul calculator PDP–1 a aparut în 1961, avea 4 K cuvinte de 18 biti si ciclul masina de 5 microsecunde. Viteza acestuia era jumatate de cea a lui IBM 7090, modelul tranzistorizat al lui IBM-709, dar costa 120.000$ fata de 1.000.000$, pretul unui IBM -7090. Astfel a aparut industria minicalculatoarelor.
Peste câtva ani a aparut PDP -8, o masina pe 12 biti, ce costa doar 16.000$.
PDP -8 avea o inventie majora: o singura magistrala, omnibus-ul, ca în figura 2.

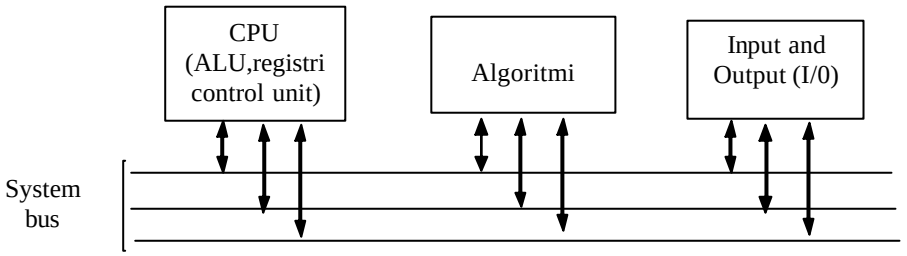

Fig. 2 Data Bus, Adres Bus, Control Bus

Figura 2 prezinta modelul bazat pe bus al unui calculator. Acest model partitioneaza un system calculator în 3 subunitati: CPU, memorie si dispozitive I/0. Acest model este de fapt o rafinare a modelului ***von Neumann*** , în care ALU si CU sunt combinate într-o singura unitate CPU.
Deasemenea unitatile I/0 sunt combinate într-o singura I/0 unit.
Mai important este modelul bus-ului sistemului, o cale comuna partajatata alcatuit din data bus (poarta informatia ce trebuie transmisa), address bus (identica unde este trimisa informatia) si control bus (descrie cum este transmisa informatia si în ce maniera). Mai exista de asemenea un power bus care nu este figurat, dar a carui prezenta este subînteleasa.
Busul de date permite transferal datelor între componentele sistemului. Unele CPU au bus-uri separate catre si dinspre CPU, caz în care avem data-in bus si data-out bus, cel mai adesea exista un singur bus între entitati ce comunica între ele, fiecare entitate trebuie sa aiba identitate

distincta–adresa. În unele computere toate adresele se presupun a fi adrese de memorie, chiar daca sunt de fapt parti ale memoriei, sau device-urile I/0, în timp ce alte computere device-urile I/O au adrese separate de I/O.
O adresa de memorie sau o locatie de memorie identifica o locatie unde data este stocata.
Revenind la istoricul calculatoarelor, în anii '60 IBM a lansat masina 7904, o forta în calculul stiintific, dar si o masina de dimensiuni reduse si cost scazut 1401.
În 1964, o noua companie Control Data Corp. (CDC) a lansat masina 6600, care era aproape de 10 ori mai rapida decât 7904. Interiorul UCP era o masina masiv paralela. 6600 avea în interior o serie de mici unitati procesoare. UCP putea sa fie astfel în întregime dedicata calculului, în timp ce managementul lucrarilor de I/0 era lasat pe seama acestor unitati procesoare. Masina 6600 implementa paralelismul de timp control-flow paralelism si putem spune ca a fost cu o decada înaintea timpului sau. Multe din ideile cheie ce stau la baza calculatoarelor moderne îsi au sursa direct în 6600.

1.1.3. A treia generatie – circuite integrate (1965 – 1980)

Circuitul integrat de siliciu a fost inventat de Robert Noyce în 1958.
Pentru înlocuirea masinilor 7094 si 1401, IBM a luat o decizie radicala în 1964. A introdus o singura linie de produse System/360, bazat pe circuite integrate, care a fost conceputa atât pentru calcule stiintifice cât si pentru calcul commercial. Astfel, au fost create 4 modele 30,40,50 si 65. 1401 putea fi înlocuit cu 360 Model 30 iar 7094 cu 360 Model 65. O alta inovatie majora a masinii 360 a fost multiprogramarea (multiprogramming), care implica existenta mai multor programe în memorie în acelasi timp, astfel încât atunci când unul asteapta pentru terminarea unei oparatii de I/0, un altul poate face calcule.
Masina 360 a fost de asemenea, prima care putea emula (simula) alte calculatoare. Modelele 30 puteau emula 1401 sau 7049 astfel încât programele binare facute pentru acestea puteau rula pe 360 fara a fi modificate.Emularea era usor de facut deoarece toate modelele initiale erau microprogramate. O alta caracteristica a masinii 360 a fost un spatiu imens de adrese la acea vreme (2^{24} octeti = 16 megaocteti).
De asemenea, un pas înainte l-a constituit a treia generatie introdusa de DEC, PDP-11. Atât 360 cât si PDP-11 aveau registre orientate pe cuvânt si o memorie orientata pe octeti.

1.1.4. A patra generatie – integrarea pe scara foarte larga (1980 -?)

Îa anii `80, integrarea VLSI (Very Large Scale Integration) a permis sa se compacteze pe un cip zeci de mii, apoi sute de mii si, în final, milioane de tranzistoare. Acest fapt a condus la aparitia calculatoarelor personale. Primele calculatoare erau vândute sub forma de kituri. Fiecare kit continea o placa de baza, câteva cipuri, incluzând si un Intel8080 sau Z80, o sursa si, eventual o unitate de disc de 8 inch. A aparut apoi sistemul de operare CP/M care a devenit destul de raspândit pe I8080.
Un alt calculator personal timpuriu a fost Apple si apoi Apple II, proiectate de Steve Jobs si Steve Wozniak.
Dupa câtva timp IBM s-a hotarât sa intre pe piata calculatoarelor personale când a constataat ca aceasta reprezinta efectiv o piata. Dar în loc sa proiecteze masina de la 0, folosind componente IBM, a însarcinat pe un director executive cu acest proiect. Acesta a dezvoltat proiectul departe de sediul IBM, în Boca Raton Florida, si a ales I8080 ca UCP a calculatorului si a construit

calculatorul personal IBM. Acesta a fost lansat în 1981 si a devenit cel mai vândut calculator din lume.
IBM a facut ceva necaracteristic pe care ulterior avea sa-l regrete. În loc sa mentina secretul proiectului noii masini sau sa o protejeze cu patente, asa cum facea în mod obisnuit, a publicat planurile compete, inclusive diagramele circuitelor într-o carte pe care a vândut-o cu 49$. Ideia de baza era dea permite altor companii sa produca placi direct încorporabile IBM PC pentru a creste flexibilitatea în popularitatea calculatorului personal. Urmarea a fost ca multe companii au început sa produca clone ale PC-ului, de multe ori la preturi mult mai mici decât IBM. S-a creat astfel o întreaga industrie.
Desi alte companii produceau PC-uri cu CPU-ri ce nu erau bazate pe microprocesoare Intel, de exemplu Commodore, Atari, Amiga, Apple, puterea IBM PC-lui a facut ca acestea sa fie scoase de pe piata. Numai o mica parte au supravetuit acoperind segmente de piata cum ar fi statiile de lucru sau supercalculatoarele.
Versiunea initiala a IBM–PC era dotata cu sistemul de operare MS-DOS produs de Microsoft Corporation. Deoarece Intel producea cipuri din ce în ce mai puternice IBM si Microsoft au dezvoltat un succesor al MS-DOS, numit OS/2 ce avea o interfata grafica similara cu cea de la Apple MacIntosh. Între timp Microsoft a dezvoltat propriul sau system de operare, Windows, ce lucra peste MS-DOS, util în cazul în care OS/2 nu s-ar fi impus. Pe scurt OS/2 nu s-a impus si Microsoft a continuat dezvoltarea sistemulu Windows, devenit astazi un standard. Modul în care o companie mica, Intel, si una si mai mica, Microsoft, au reusit sa detroneze gigantul IBM, una din cele mai puternice si mai bogate companii din istorie, ramâne un fapt remarcabil.
De la mijlocul anilor ’80 a început sa se impuna o structura numita RISC (Reduced Instrucsion Set Computer) care sa înlocuiasca arhitecturile CISC complicate cu unele mai simple si mai rapide. Astfel de masini puteau sa execute mai multe instructiuni în acelasi timp. Conceptele RISC, CISC si superscalar vor fi introduce în alt capitol.

1.2. Organizarea structurata a calculatoarelor

1.2.1 Limbaje, niveluri si masini virtuale

Calculatoarele utilizeaza un set de instructiuni masina. Acestea formeaza împreuna un limbaj pe care îl vom numi L0. Programele scrise de utilizator sunt într-un limbaj L1, mult mai accesibil decât limbajul masinii.
O metoda posibila de executare a unui program scris în L1 este aceea de a înlocui fiecare instructiune din acest limbaj cu o secventa echivalenta de instructiuni L0.
Programul rezultat este format numai din instructiuni scrise numai în limbajul L0. O astfel de tehnica se numeste traducere (translation).
Cea de-a doua tehnica consta în a scrie un program în L0 care preia programe din L1 ca date de intrare si le executa examinând fiecare instructiune pe rând si executând secventa echivalenta de instructiuni din L0. O stfel de tehnica se numeste interpretare (interpretation), în programul care o realizeaza se numeste interpretor.
Traducerea si interpretarea sunt similare.
În loc de a gândi în termini de traducere si interpretare de multe ori este mai simplu sa ne imaginam existenta unui calculator virtual sau masina virtuala (virtual machine) al carui limbaj masina este L1. Sa numim aceasta masina virtuala M1 si sa numim masina viruala corespunzatoare lui L0 cu M0. Daca o astfel de masina ar putea fi construita la un pret rezonabil

nu am mai putea avea nevoie de L0 sau de o masina care sa execute programe în L0. Programatorii ar putea scrie programele în L1 si le-ar putea executa direct. Dar masina virtuala ce suporta limbajul L1 este prea scumpa sau prea complicata pentru a putea fi realizata fizic. Totusi oamenii au posibilitatea sa scrie programe pentru aceasta. Aceste programe vor fi interpretatae sau traduse de un program scris în L1 care poate fi executat pe calculatorul existent. Pe scurt, pot fi scrise programe pentru masini virtuale ca si cum acestea ar exista în realitate.
Pentru ca traducerea si interpretarea sa fie practica limbajele L1 si L0 nu trebuie sa fie "prea" diferite. Acesta înseamna ca L1, desi mai bun ca L0 va fi nepotrivit pentru multe aplicatii. Abordarea evidenta este de a inventa un alt set de instructiuni mai prietenos cu utilizatorul si mai putin orientat masina ca L1. Acest al treilea set formeaza un nou limbaj pe care îl numim L2 (cu masina virtuala corespunzatoare M2). Programatorii pot scrie programe în L2, ca si cum o masina virtuala cu limbajul L2 ar exista în realitate. Astfel de programe pot fi traduse în L1 sau, alternativ, executate cu un interpretor scris în L1.
Inventarea unei serii întregi de limbaje, fiecare mai convenabil decât predecesoarele poate continua pâna la crearea unui limbaj potrivit. Fiecare limbaj foloseste predecesorul sau ca baza, deci, folosind aceasta tehnica putem vedea calculatorul ca o serie de straturi (layers) sau nivele unul deasupra celuilalt ca în figura 3.

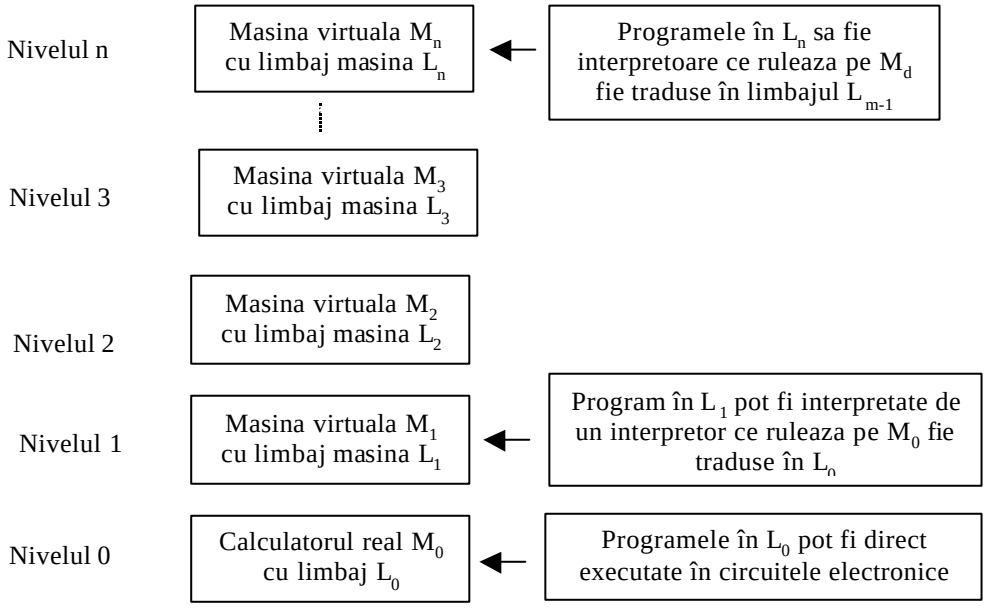

Figura 3.

Un calculator cu n niveluri poate fi vazut ca în masini virtuale diferite, fiecare cu un limbaj masina diferit. Numai programele scrise în L0 pot fi direct executate de circuitele electronice, fara a necesita o traducere sau interpretare.
Un programator ce a scris un program pentru masina virtuala de nivel n nu trebuie sa se preocupe de interpretarile si translatarile de dedesubt. Cu mai multi utilizatori ce folosesc o masina de nivel n sunt interesati de ultimul nivel, deci cel care seamana cel mai putin cu limbajul masina de pe nivelul de baza. Cei care sunt interesati sa înteleaga cum functioneaza de fapt un calculator trebuie sa studieze toate nivelurile. Cei interesati de proiectarea unui calculator sau a unui nou nivel trebuie de asemenea, sa fie familiarizati si cu alte niveluri decât cele de sus.

1.2.2. Masini multi–nivel contemporane

Cele mai multe calculatoare moderne sunt formate din doua sau mai multe niveluri. Exista pâna la 6 niveluri ca în Figura 4. Nivelul 0 corespunde structurii hardware a masinii. Circuitele executa programele în limbaj – masina de pe nivelul 1. Trebuie mentionata existenta unui nivel suplimentar, aflat sub nivelul 0. Acest nivel, care nu apare în figura, tine de domeniul ingineriei electrice si se numeste nivelul echipamentelor (device level).
La nivelul cel mai de jos avem nivelul logic digital, format din porti. Desi sunt alcatuite din tranzistoare, portile pot fi modelate cu acuratete ca circuite digitale. Fiecare poarta are una sau mai multe intrari digitale (0 sau 1 logic) si calculeaza, în functie de acestea o valoare de iesire pe baza unei functii scumpe cum ar fi SI, SAU, NU. Câteva porti combinate pot forma o memorie de un bit, ce poate stoca 0 sau 1.
Memoriile de un bit pot fi combinate în grupuri de 16, 32 sau 64 (de ex.) pentru a forma registre. Fiecare registru contine un numar binar pâna la o anumita valoare.
Urmeaza nivelul microarhitecturii. La acest nivel avem în mod obisnuit o colectie de 8 pâna la 32 de register care formeaza o memorie locala si un circuit UAL ce poate executa operatii aritmetice simple. Registrele sunt conectate la UAL pentru a forma o cale de date (data path) prin care se transmit datele. Operatiile principale ale caii de date constau în selectia a 1 sau 2 registri asupra carora actioneaza UAL, de exemplu le aduna si rezultatul este stocat apoi în unul din registrii.

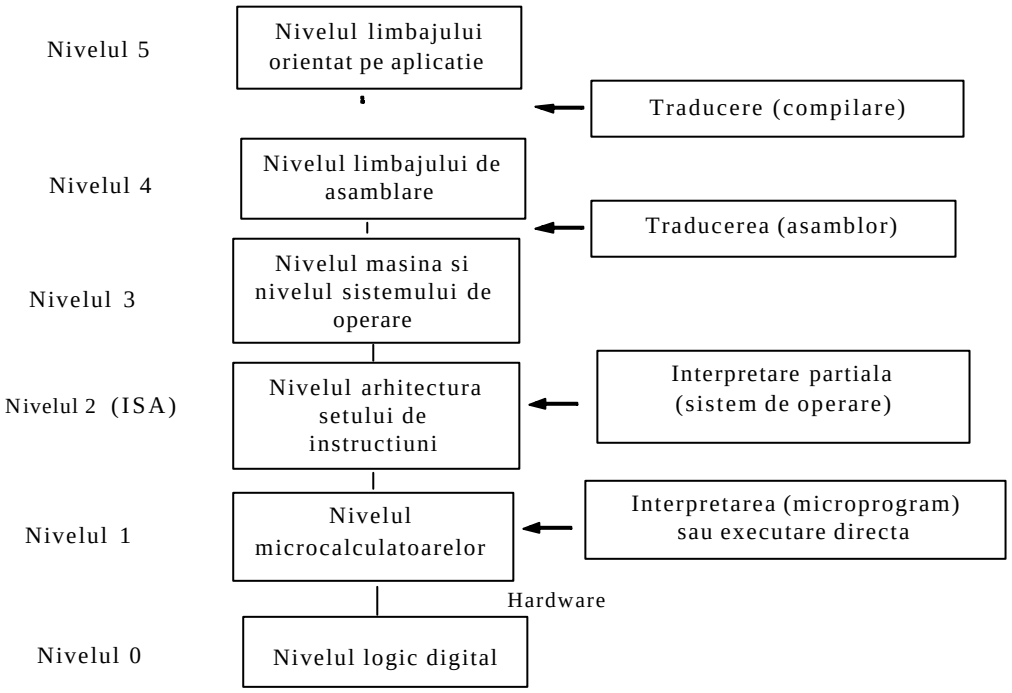

Figura 4

Pe unele masini operatiile caii de date sunt controlate de un program numit microprogram. Pe alte masini calea de date este controlata direct prin hardware.

Pe masinile cu control software al caii de date, microprogramul este un interpretor ale instructiunilor de pe nivelul 2. Acesta extrage, examineaza si executa instructiunile secvential folosind pentru aceasta calea de date. De exemplu pentru ADD se va extrage intructiunea, se vor localiza si aduce în registru operanzii, UAL calculeaza suma si în final rezultatul va fi depus într-un registru. Pe o masina cu control hardware al caii de date se vor executa pasii similari dar fara ajutorul unui program memorat explicit pentru controlul interpretarii instructiunilor de pe nivelul 2.

Nivelul 2 este nivelul arhitecturii setului de instructiuni (sau nivelul ISA – Instruction Set Arhitecture). Fiecare producator de calculatoare publica un manual ce cuprinde si acest set de instructiuni. Aceste manuale trataeza nivelul ISA, dar si nivelele de dedesupt.

Urmatorul nivel este de obicei un nivel hibrid. Cele mai multe din instructiuni sunt de obicei, de asemenea, intructiuni de nivelul ISA. În plus exista un set de instructiuni noi, o organizare diferita a memoriei, posibilitatea de executare concurenta a doua sau mai multe programe si diverse alte caracteristici. Exista mai multe posibilitati de proiectare a nivelului 3 decât în cazul nivelurilor 1 si 2.

Noile facilitati adaugate pe nivelul 3 sunt realizate de un interpretor care se executa de pe nivelul 2, interpretor numit sistem de operare din motive istorice. Instructiunile de pe nivelul 3 ce sunt identice cu cele de pe nivelul 2 sunt executate direct de microprogram (sau controlate hardware) si nu de sistemul de operare. Astfel spus, unele instructiuni de pe nivelul 3 sunt interpretate de sistemul de operare, iar alte instructiuni de pe nivelul 3 sunt interpretate direct de microprogram. Aceasta este semnificatia termenului "hibrid" folosit anterior. Vom numi acest nivel, nivelul masina al sistemului de operare.

Primele trei niveluri ne sunt proiectate a fi utilizate de programatorul obisnuit. Ele sunt concepute în principal pentru executia interpretoarelor si translatoarelor necesare sustinerii nivelurilor superioare. Aceste interpretoare si translatoare sunt scrise de programatorii de sitem (putini la numar) care sunt specializati în proiectarea si implementarea de noi masini virtuale. Nivelul 4 si cele superioare sunt destinate programatorilor de aplicatii.

O alta modificare ce apare pe nivelul 4 este metoda prin care sunt sustinute nivelurile superioare. Nivelurile 2 si 3 sunt întodeauna interpretate . Nivelurile 4 si 5 si alte superioare sunt de obicei sustinute de translatoare, desi nu întotdeauna.

O alta diferenta între nivelele 1,2,3 si 4,5 este natura limbajului oferit. Limbajele masina ale nivelurilor 1,2,3 sunt numerice. Începând cu nivelul 4 limbajele contin cuvinte sau abrevieri pe întelesul oamenilor.

Nivelul 4, nivelul limbajului de ansamblare, este de fapt o forma simbolica pentru unul din limbajele inferioare. Acest limbaj ofera utilizatorilor o metoda de o serie de programe pentru nivelurile 1, 2, 3 într-o forma care nu este chiar asa de dificila ca cea a limbajelor virtuale. Programul care efectueaza traducerea de pe acest nivel se numeste asamblor (assembler).

Nivelul 5 contine de obicei limbaje proiectate pentru afi utilizate pentru programatori de aplicatii. Aceste limbaje sunt frecvent nimite limbaje de nivel înalt**.** Exista sute de astfel de limbaje. Dintre cele mai cunoscute se numara C, C++, BASIC, JAVA, LISP, PROLOG, PASCAL. Programele scrise în aceste limbaje sunt traduse pentru nivelurile 3, 4 de translatoare cunoscute sub numele de compilatoare, desi acestea pot fi uneori interpretate. De exemplu., programele scrise în Java sunt deseori interpretate. În unele cazuri nivelul 5 reprezinta un interpretor pentru un domeniu aplicativ, particular, de exemplu calcul simbolic.

În esenta calculatoarele sunt proiectate ca o serie de niveluri, fiecare nivel fiind construit pe baza precedecesorului sau. Fiecare nivel reprezinta o abstractizatre distincta cu obiecte si operatii

diferite. Multinea de tip uri de date, operatii si caracteristici ale fiecarui nivel se numeste arhitectura. Arhitectura se refera la aspecte care sunt vizibile pentru utilizatorulul uni nivel. Diferite caracteristici vazute de programator, de exemplu memoria existenta, sunt parti ale arhitecturii. Aspectele de complementare, de exemplu tehnologia de realizare a cipurilor de memorie nu fac parte din arhitectura (ci din organizare). În practica comuna arhitectura calculatoarelor si organizarea calculatoarelor înseamna acelasi lucru.

1.2.3. Evolutia masinilor multinivel

Circuitele electronice împreuna cu memoria si dispozitivele I/0 formeaza partea hardware a calculatorului. Hardware e format din obiecte targibile. Partea software consta din algoritmi si reprezentarea acestora în calculator. Programele pot fi memorate pe discuri de masa, disccuri flexibile, CD-rom si alte suporturi, dar esenta software este setul de instructiuni ce formeaza programul si nu mediile fizice pe care acestea sunt memorate.
În cazul primelor calculatoare exista o diferenta clara între hardware si software. În timp aceasta diferenta s-a estompat considerabil datorita elaborarii eliminarii sau compactarii unor niveluri o data cu evolutia calculatoarelor.
Orice operatie executata prin software poate fi foarte bine construita direct în hardware, de preferinta, dupa ce este destul de bine înteleasa. Si reversul este adevarat: orice instructiune executata hardware poate fi simulata prin soft. Decizia de a realize anumite functii hardware si altele software depinde de factori cum sunt: costul, viteza ,siguranta în funtionare si evolutia viitoare.
Microprogramarea. Primele calculatoare digitale din anii '40 aveau doar doua niveluri: nivelul ISA în care se facea toata programarea si nivelul logic digital.
În 1951 Maurice Wilkes (Cambridge University) a avut ideia proiectarii unui calculator cu trei niveluri. Acesta trebuia sa aiba un interpretor cablat nemodificabil (microprogramul) ale carui functii trebuiau sa execute programul de pe nivelul ISA prin interpretare. Deoarece în acest caz hard-ul trebuia sa execute numai microprograme, cu un set limitat de instructiuni, era nevoie de mai putine circuite electronice. Multe masini de acest tip s-au construit în ani 60 si 70(de exemplu PDP 11 al DEC). În anii `70 ideia ca nivelul ISA sa fie interpretat de un microprogram a devenit dominanta.

Aparitia sistemului de operare

Acestea au aparut în anii `60 pe sietemele batch processing. Ideia a fost sa exista un program numit sistem de operare rezindent în calculator. Acesta a fost gândit în scopul usurarii sarcinii operatorilor si pentru a micsora timpii morti. În anii urmatori aceste programe s-au dezvoltat pâna când au început sa semene cu un nou nivel. Câteva din instructiunile acestui nou nivel erau identice cu instructiunile de pe nivelul ISA, dar altele în special prin structiunile de I/0 erau complet diferite. Aceste noi instructiuni erau numite macrou-ri ale sistemului de operare sau apeluri de supervisor. Termenul uzual în present este apel de system (system call).

Migrarea catre microcod si eliminarea microprogramarii

O data cu microprogramarea, proiectantii si-au dat seama ca pot adauga noi instructiuni numai prin extinderea microprogramului. Deci pot adauga “hardware” (noi instructiuni masina)

prin programare. Acest fapt a condus la crestere consistenta a setului masina, aparând pe acelasi calculatoare seturi de instructiuni din ce în ce mai mari puternice.

De exemplu unele masini aveau o instructiune INC (aduna unu la un numar). Pentru ca masinile aveau si o instructiune ADD generala, existenta acestei instructiuni (ADD +1) nu era necesara. Totusi instructiunea INC s-a impus, fiind de obicei mai rapida decât ADD.

Multe din instructiunile adaugate includeau adesea

-instructiuni pentru înmultirea / împartire întregi;
- instructiuni pentru aritmetica în virgula mobila;
- instructiuni pentru apelul si revenirea din procedeuri;
- instructiuni pentru cresterea vitezei ciclurilor;
- instructiuni pentru tratarea sirurilor de caractere.

Apoi au început sa caute alte instructiuni pe care sa le introduca în microprograme:

- instructiuni pentru calcul matricial;
- caracteristici ce permit programelor sa fie mutate în memorie dupa ce au început sa ruleze;
- sistem de întreruperi ce activeaza calculul de îndata ce operatia I/0 a fost terminata;
- posibilitatea de a suspenda un program si de a începe altul.

Microprogramele au devenit din ce în ce mai mari prin anii `70 si aveau tendinta sa devina din ce în ce mai lente pe masura ce au acumulat mai multe facilitati. Între timp se putea creste viteza proceselor prin eliminarea microprogramarii, reducerii drastica a setului de instructiuni si executarea setului ramas direct pe masina (controlul hardware al caii de date).

Toate acestea demonstreaza ca generatiile dintre hardware si software este fluida, arbitrara si în continua schimbare. Multe dintre delimitarile între nivelurile prezentate anterior sunt fluide si reflecta situatia din prezent. Dar din punct de vedere al programatorului nu este important cum este de fapt implementata o instructiune. Programând în ISA, instructiunea de înmultire poate fi utilizata ca si cum ar fi o instructiune hardware, fara a ne preocupe cum este executata. Ce este hardware pentru o persoana poate fi software pentru alta. Vom reveni la aceasta discutie.